Fakulteta ta računalništvo in informatika, Univerza v Ljubljani

Poročilo

Delo pri predmetu Razvoj inteligentnih sistemov

Jaka Jenko - 63160145

Žan Žagar - 63160340

Žan Pristopec -

Mentor: Daniel Skočaj

Asistent:

Kazalo

No table of contents entries found.

# Uvod

//kaj je bila naloga

# Metode

//kako smo kaj naredili - teoretično

## Izdelava mape

Za premikanje robota po poligonu, je bila prva in najbolj pomembna stvar, izdelava ustrezne mape poligona, po kateri se je robot premikal.

Prvi korak je bil, da smo na robotu vklopili ustrezne procese, za izrisovanje mape na rviz, potem smo robota s pomočjo turtlebot procesov manevrirali po prostoru, da se nam je "izrisovala" mapa na rvizu.

Zaradi narave senzorjev, smo morali to večkrat ponoviti, dokler nismo dobili rezultatov, ki so se nam zdeli sprejemljivi, torej dobili smo sliko mape, ki je zgledala približno prava.

To mapo smo potem polepšali z GIMP-om (odstranili šum in čudna zaznavanja), in jo potem uporabljali pri kasnejših nalogah, kjer je to bilo zahtevano.

## Premikanje robota

//JAKA

## Zaznavanje krogov

//JAKA

## Zaznavanje cilindrov

//JAKA

## Premikanje robotske roke

V sklopu 6 naloge (exercise 6), je bilo zahtevano delo z robotsko roko. Robotska roke se upravlja tako, da pišemo pozicije določenih servotov, ki jih potem sprejema nek subscriber, kar nam je nalogo malce utežilo je to, da je vsaka robotska roka kalibrirana malo drugače.

V ta namen smo ustvarili skripte, ki nam glede na podane min in max vrednosti rok na posameznih robotih pretvori vrednosti ene robotske roke v procente, ki jih potem lahko z drugo skripto uporabimo, da dobimo dejanske vrednosti pozicij, na drugem robotu z drugo roko.

S to skripto lahko v relativno majhnem času izračunamo iz starih "navodil" (skupek ukazov za premikanje posameznih servo motorjev), nova navodila, ki so uporabna na drugem robotu z drugo roko, in imajo isto funkcijo.

Dejanska implementacija pobiranja in odlaganja kovanca je pa samo neka zanka, ki vklučuje navodila za servo pozicije (publishanje na topic premika servo motorjev), in iz neke startne pozicije vzeme kovanec, ga odvrže, in se vrne v startno pozicijo.

## Prepoznavanje številk

//PRISTOPEC

## Prepoznavanje QR-kod

Ker je bila skripta za detekcijo QR kode iz krogov že podana, je bil najtežji del tega sklopa najti del kode, kjer bi vključili zaznavanje. QR kodo zaznamo tako, da vzamemo sliko kroga, ki se poravna na "okvir" (okvir so v tem primeru 4 manjše qr kode, ki jih skripta detektira), potem pa iz centra slike poskušamo prebrati QR kodo, in jo z knjižnico pyzbar dešifrirati, in izpisati njeno vrednost.

## Prepoznavanje grafov

//PRISTOPEC

## Prepoznavanje barv

Za prepoznavanje barv smo uporabili skripto za detekcijo elips, ta skripta, je od centra zunanje in notranje elipse odštela vertikalen in horizontalen polmer, da smo potem na ta način dobili 4 točke, "v krogu", torej med zunanjo in notranjo elipso, ta del je bil pobarvan.

Od teh 4 točk smo prebrali RGB in HSV vrednosti, in jih zapisali v .csv datoteko, podali smo tudi razred, torej dejansko barvo teh 4 točk. V to .csv datoteko, smo tudi dodali dodatne odtenke barv, zaradi tega da je bila podatkovna množica večja.

To smo naredili za vsak krog, kar nam je dalo .csv file z približno 100 vnosi. Na podlagi te datoteke smo zgradili klasifikacijsko drevo v programu R, in ga testirali na testni množici, z njem smo dosegli 100% točnost na krogih in cilindrih v poligonu.

To drevo smo z if stavkim vgradili v python skripto, in v c++ program.

V python skripti na enak način vzamemo 4 točke od zaznanih krogov, in za vsako točko z if stavkom ugotovimo predviden razred (red,green,yellow,blue), in potem vzamemo najbolj predviden razred (recimo 3 točke zelena, 1 rumena = razred = zelena).

V cylinder segmentation datoteki pa vzamemo celoten point cloud cylinder objekt, z RGB vrednostimi (PointXYZRGB) in iteriramo po celotnem objektu, ter seštevamo vrednosti R G in B vseh točk, ter jih povprečimo, tako potemo z pomočjo Marker publisherja pošljemo vrednost barve (R G in B normiran na razpon 0.00 do 1.00, ter jo prejmemo na drugi strani, na glavni datoteki. Tam potem iz RGB z pomočjo funkcije izračunamo še HSV vrednost povprečja cilindra, in ga z istim klasifikatorjem kot zgoraj skušamo dati v razred red,green,blue ali yellow. Tudi tukaj smo imeli 100% točnost v poligonu.

Barve in XY pozicije krogov in cilindrov si v glavni datoteki zapovnemo, in lahko tako preverjamo če je že novo detektiran krog/cilinder že bil detektiran.

# Implementacija

//kako smo vse skupaj povezali

//JAKA

# Rezultati

## Prepoznavanje barv

Na cilindrih in krogih v poligonu, smo z uporabo našega klasifikacijskega drevesa, opisanega zgoraj, ki klasificira barve v razrede: red,green,yellow in blue dosegli 100% točnost. Je bila velik faktor osvetljenost, in v primeru zelo svetlo zelene barve (takšne NI v poligonu), je možnost da jo napačno klasificira kot rumeno.

## Prepoznavanje številk, QR-kod, grafov

//poda kakimi koti in razdaljami ti zaznava zgornje zadeve

//PRISTOPEC

# Razdelitev dela

## Jaka Jenko

//JAKA

## Žan Žagar

Funkcije, ki hranijo podatek o zaznanih krogih, cilindrih, prek marker objektov, da nebi dvakrat zaznali istega kroga ali cilindra.

Funkcije za dvigovanje in spuščanje kovancev v cilinder, ter skripte za lažjo kalibracijo na drugih robotih.

Gradnja podatkovne množice za detekcijo barve krogov in cilindrov.

Učenje in gradnja klasifikatorja na zgoraj omenjeni podatkovni množici barv krogov in cilindrov.

Zaznavanje barv krogov in cilindrov z klasifikacijskem drevesu (pomagal Jaka Jenko).

Funkcija za izpis vseh markerjev v RVIZ.

## Žan Pristopec

//PRISTOPEC

# Zaključek

//JAKA